Assignment 6-7 Polynomial Multiplication 4

You are given two polynomials $f(x)$ and $g(x)$ with integer coefficients. In this problem you'll have to find out the product $f(x) \cdot g(x)$ of $f(x)$ and $g(x)$. For the sake of convenience, we define the zero polynomial as $0x^0$.

Input

The input consists of t ($30 \le t \le 40$) test cases. The first line of the input contains only positive integer t. Then t test cases follow. Each test case consists of two polynomials $f(x)$ and $g(x)$. Each polynomial consists of three lines: the first line contains only one positive integer m ($m \le 10$) which represents the number of nonzero terms of the polynomial; the second line consists of m nonzero integers (in the range $[-2^{25}, 2^{25} - 1]$) representing the coefficients of the polynomial; the third line consists of m integers (in the range $[0, 2^{62} - 1]$ and in decreasing order) representing the corresponding exponents of the polynomial. For example, the polynomial $300x^{1000} + 200x^{100} + 100x^{10}$ is represented as the following three lines:

```
3
300 200 100
1000 100 10
```

You may assume that neither $f(x)$ nor $g(x)$ is the zero polynomial.

Output

For each test case, you are to output three lines representing $f(x) \cdot g(x)$, in the same format as the input.

Sample Input

```
2
3
1 -2 -4
3 2 0
1
4
0
3
-1 -2 4
3 2 0
4
1 2 1 -7
```

3 2 1 0

Sample Output

3
4 -8 -16
3 2 0
7
-1 -4 -5 9 22 4 -28
6 5 4 3 2 1 0

Requirements

In your program, a polynomial $a_n x^{d_n} + a_{n-1} x^{d_{n-1}} + \cdots + a_2 x^{d_2} + a_1 x^{d_1} + a_0 x^{d_0}$ should be represented by the following two arrays:

	n	$n-1$	$\cdots$	2	1	0
coefficient array:	a_n	a_{n-1}	$\cdots$	a_2	a_1	a_0

	n	$n-1$	$\cdots$	2	1	0
exponent array:	d_n	d_{n-1}	$\cdots$	d_2	d_1	d_0

For example, the polynomial $-x^3 - 2x^2 + 4$ is represented by the following two arrays:

	2	1	0
coefficient array:	-1	-2	4

	2	1	0
exponent array:	3	2	0

Note that $d_n > d_{n-1} > \cdots > d_2 > d_1 > d_0 \geq 0$, and for every $i = 0, 1, \ldots, n$, $a_i \neq 0$.

Part of the program

You are required to write the function `addition` and `multiplication` to complete the following program which solves this problem. In your program, you cannot declare global variables or static arrays.

```
// Polynomial multiplication
#include <iostream>
using namespace std;

// product = multiplicand * multiplier
void multiplication( long long int *multiplicandCoef, long long int
*multiplicandExpon, int multiplicandSize,
                     long long int *multiplierCoef, long long int
*multiplierExpon, int multiplierSize,
                     long long int *&productCoef, long long int *&productExpon,
```

```
int &productSize );

// addend += adder
void addition( long long int *&addendCoef, long long int *&addendExpon, int
&addendSize,
               long long int *adderCoef, long long int *adderExpon, int
adderSize );

// returns true if and only if the specified polynomial is the zero polynomial
bool isZero( long long int *coefficient, int size );

// outputs the specified polynomial
void output( long long int *coefficient, long long int *exponent, int size );

int main()
{
   int T;
   cin >> T;
   for( int t = 0; t < T; t++ )
   {
      int multiplicandSize;
      cin >> multiplicandSize; // input multiplicand
      long long int *multiplicandCoef = new long long
int[ multiplicandSize ]();
      long long int *multiplicandExpon = new long long
int[ multiplicandSize ]();
      for( int i = multiplicandSize - 1; i >= 0; i-- )
         cin >> multiplicandCoef[ i ];
      for( int i = multiplicandSize - 1; i >= 0; i-- )
         cin >> multiplicandExpon[ i ];

      int multiplierSize;
      cin >> multiplierSize; // input multiplier
      long long int *multiplierCoef = new long long int[ multiplierSize ]();
      long long int *multiplierExpon = new long long int[ multiplierSize ]();
      for( int i = multiplierSize - 1; i >= 0; i-- )
         cin >> multiplierCoef[ i ];
      for( int i = multiplierSize - 1; i >= 0; i-- )
         cin >> multiplierExpon[ i ];

      int productSize;
      long long int *productCoef;
      long long int *productExpon;
      // product = multiplicand * multiplier
      multiplication( multiplicandCoef, multiplicandExpon,
multiplicandSize,
                      multiplierCoef, multiplierExpon, multiplierSize,
                      productCoef, productExpon, productSize );

      output( productCoef, productExpon, productSize );

      delete[] multiplicandCoef;
      delete[] multiplicandExpon;
      delete[] multiplierCoef;
      delete[] multiplierExpon;
      delete[] productCoef;
      delete[] productExpon;
   }
}

// product = multiplicand * multiplier
void multiplication( long long int *multiplicandCoef, long long int
*multiplicandExpon, int multiplicandSize,
                     long long int *multiplierCoef, long long int
*multiplierExpon, int multiplierSize,
                     long long int *&productCoef, long long int *&productExpon,
int &productSize )
{
```

```
}

// addend += adder
void addition( long long int *&addendCoef, long long int *&addendExpon, int
&addendSize,
               long long int *adderCoef, long long int *adderExpon, int
adderSize )
{









}

// returns true if and only if the specified polynomial is the zero polynomial
bool isZero( long long int *coefficient, int size )
{
   return ( size == 1 && coefficient[ 0 ] == 0 );
}

// outputs the specified polynomial
void output( long long int *coefficient, long long int *exponent, int size )
{
   cout << size << endl;
   cout << coefficient[ size - 1 ];
   for( int i = size - 2; i >= 0; i-- )
      cout << " " << coefficient[ i ];
   cout << endl;

   cout << exponent[ size - 1 ];
   for( int i = size - 2; i >= 0; i-- )
      cout << " " << exponent[ i ];
   cout << endl;
}
```